Software–Projekt 2007/08

VAK 03-05-G-901.01

JAKARTA

Software Development

Architekturbeschreibung

Artur Malek	amalek@tzi.de	2152026
Yasin Ünsal	yasinu@tzi.de	1874544
Levent Özenen	levent85@tzi.de	2131928
Sascha Schmidt	rone1983@web.de	2174364

Abgabe: 20. März 2008

Inhaltsverzeichnis

0 Version und Änderungsgeschichte

(Levent)

0.1 Version

Öffentliche Version 2.0
Interne Version 1.3

0.2 Änderungsgeschichte

Version	Änderungen
1.0	Die erste Veröffentlichung
1.1	Modulsicht neu
1.2	Ausführungssicht neu
1.3	konzeptionelle Sicht neu
2.0	Zweite Veröffentlichung

1 Einführung

(Levent)

1.1 Zweck

Dieses Dokument beschreibt die Architektur des Systems. Die Architektur dient dazu, zu beschreiben, wie die Anforderungen, die in der Anforderungsspezifikation angegeben sind, realisiert werden können.

Entwickler	**Tester**
dient den Entwicklern als Vorgabe für die Implementierung	dient den Testern als Grundlage für die Entwicklung von Schnittstellentests

1.2 Status

Unser Dokument beschreibt den ersten Entwurf. Es wurde noch nicht durch ein Architektur-Review freigegeben.

1.3 Definitionen, Akronyme und Abkürzung

GUI: Graphical User Interface
Sie ist eine grafische Benutzeroberfläche von Computerprogrammen.
Linux: Sie ist ein freies und portables Betriebssystem. Dieser Quelltext ist frei der Öffentlichkeit zugänglich. Der Finne Linus Torvalds hat Linux entwickelt.
Unix: ist der Oberbegriff für alle auf dem ursprünglichen, von AT und T entwickelten Unix-Entwurf basierenden Betriebssysteme, wie z.B. Linux, HP-UX etc.
UML: Unified Modeling Language –ist der Name der zur Modellierung verwendeten Technik.
PDA: PDA bedeutet Personal Digital Assistant und ist ein kleiner Computer, auf dem Computer sind die Anwendungen wie z.B. ein Adressbuch, ein Terminplaner, ein Kalender, ein Notizblock, installiert. Der PDA lässt sich über einen Touch-Screen und einige Hardware-Tasten bedienen. PDAs haben wenig Speicher, der sich allerdings durch Speicherkarten erweitern lässt.

1.4 Referenzen

Vorlesung Software-Projekt Universität Bremen 07/08
http://www.informatik.uni-bremen.de/st/swp

1.5 Übersicht über das Dokument

In Kapitel 2.1 wird man zunächst auf die Einflussfaktoren für den Entwurf, Probleme und deren mögliche Lösungsstrategien eingehen. Anschliessend werden in den Kapiteln 3, 4 und 5 drei der vier Sichten nach Hofmeister beschrieben.Kapitel 6 wird geklärt, wie das System geändert werden muss, wenn sich evtl. die Rahmenbedingungen oder die Anforderungen ändern.

2 Globale Analyse

2.1 Einflussfaktoren

2.1.1 Organisation

(bearbeitet von: Yasin Ünsal)

Managment		
OR 1.0	**Umgebung**	
OR 1.01	**Universität Bremen**	
Flexibilität	**Änderbarkeit**	**Auswirkungen**
hoch - Variable Arbeitsumgebungen vorhanden	gering -	auf : OR 1.1, OR 1.2

Managment		
OR 1.1	**Organisation Projektgruppe**	
OR 1.11	**Projektaufteilung in 5 Phasen**	
Flexibilität	**Änderbarkeit**	**Auswirkungen**
hoch - Umplanung jederzeit möglich	gering - Änderung der Projektgrundlage	auf : OR 1.0, OR 1.2, OR 2.0, OR 4.1, OR 5.0

Managment		
OR 1.2	**Kommunikation Projektgruppe**	
OR 1.21	**Elektronisch (Messenger, Mail) und Persöhnliches treffen**	
Flexibilität	**Änderbarkeit**	**Auswirkungen**
hoch - Große wahl zwischen Kommunikationsmöglichkeiten	gering - Die gewählten Kommunikationstechniken sind Komfortabel	auf : OR 4.1

Personal		
OR 2.0	**Anforderungen an das Personal**	
OR 2.01	**Verständniss der Aufgaben**	
Flexibilität	**Änderbarkeit**	**Auswirkungen**
gering - Verwechseln der Aufgabenbereiche Möglich	gering -	auf : OR 1.1, OR 4.1

Prozeß- und Entwicklungsumgebung		
OR 3.0	**Entwicklungswerkzeuge**	
OR 3.01	**NetBeans**	
Flexibilität	**Änderbarkeit**	**Auswirkungen**
hoch - Änderung des Werkzeugs möglich	gering - Add-ons verfügbar	auf : OR 2.0, OR 3.1

Prozeß- und Entwicklungsumgebung		
OR 3.1	**Betriebssystem**	
OR 3.11	**Windows / Linux**	
Flexibilität	**Änderbarkeit**	**Auswirkungen**
mittel - Umgebungswechsel	mittel - Änderung des Betriebssystem möglich (eingeschränkt)	auf : OR 3.0, OR 3.2, OR 2.0

Prozeß- und Entwicklungsumgebung		
OR 3.2	**Werkzeuge des Änderungsmanagments**	
OR 3.21	**SVN Tortoises**	
Flexibilität	**Änderbarkeit**	**Auswirkungen**
keine - nicht möglich	hoch - Andere Programme verfügbar	auf : OR 2.0, OR 3.1, OR 3.0

Prozeß- und Entwicklungsumgebung		
OR 3.3	**Entwicklungsplattformen**	
OR 3.31	**JAVA SDK**	
Flexibilität	**Änderbarkeit**	**Auswirkungen**
gering - Update verfügbar	gering - Programm-Code kann in andere Sprachen angepasst werden	auf : OR 2.0, OR 3.1, OR 3.0, OR 5.0, T 1.3

Prozeß- und Entwicklungsumgebung		
OR 3.4	**Testprozesse**	
OR 3.41	**Blackbox- und Whitebox-test**	
Flexibilität	**Änderbarkeit**	**Auswirkungen**
hoch - Kundenspezifisch	gering - andere Testmethoden	auf : OR 4.1, OR 5.0, OR 2.0

Entwicklungszeitplan		
OR 4.0	**Time-to-Market**	
OR 4.01	**Juli 2008**	
Flexibilität	**Änderbarkeit**	**Auswirkungen**
keine - Auslieferungstermin steht fest	gering - Nach Absprache mit dem Kunden möglich	auf : OR 4.1, OR 5.1

Entwicklungszeitplan		
OR 4.1	**Zeitplan**	
OR 4.11	**GANTT-Diagramm**	
Flexibilität	**Änderbarkeit**	**Auswirkungen**
mittel - Zeitplan wird dynamisch angepasst	gering - Durch vielzahl von Gründen	auf : OR 4.0, OR 5.1, OR 3.4, OR 1.0, OR 1.1, OR 1.2

Entwicklungsbudget		
OR 5.0	**Anzahl der Mitarbeiter**	
OR 5.01	**5 Mitarbeiter**	
Flexibilität	**Änderbarkeit**	**Auswirkungen**
mittel - Es können hilsarbeiter hinzugezogen werden	mittel - Mitarbeiter können ausfallen	auf : OR 1.0, OR 1.1, OR 1.2, OR 4.1, OR 5.1, OR 2.0

Entwicklungsbudget		
OR 5.1	**Budget**	
OR 5.11	**Menschliche Ressourcen**	
Flexibilität	**Änderbarkeit**	**Auswirkungen**
mittel - Hilfe 3.Personen möglich	hoch - Wissen erweitern durch lernen	auf : P 1.0

2.1.2 Produkt

Produktfaktoren		
P 1.0	**Produktkosten**	
P 1.01	**Kostenlos**	
Flexibilität	**Änderbarkeit**	**Auswirkungen**
keine - Vertraglich festgelegt	gering - Durch Kunden vereinbart möglich	auf : OR 5.1

Produktfaktoren		
P 1.1	**Plattformunabhängigkeit**	
P 1.11	**JAVA VM**	
Flexibilität	**Änderbarkeit**	**Auswirkungen**
keine - festgelegt	keine - festgelegt	auf : T 1.4

Produktfaktoren		
P 1.2	**Datenschutz**	
P 1.21	**Gesetzliche Vorgaben**	
Flexibilität	**Änderbarkeit**	**Auswirkungen**
keine - festgelegt	gering - Durch Kundenwünsche	auf : keine

Produktfunktionen		
P 2.0	**Benutzerschnittstelle**	
P 2.01	**s.Prototyp**	
Flexibilität	**Änderbarkeit**	**Auswirkungen**
gering - vorgabe durch Prototyp	gering - Durch Kundenwünsche	auf : keine

Produktfunktionen		
P 2.1	**Fehlererkennung**	
P 2.11	**Übergeben an den Webserver**	
Flexibilität	**Änderbarkeit**	**Auswirkungen**
gering - Fehler in Log datei speichern	mittel - Neue Funktionen	auf : OR 4.1

Produktfunktionen		
P 2.2	**Sicherheit**	
P 2.21	**Verschlüsselung des Datenverkehrs**	
Flexibilität	**Änderbarkeit**	**Auswirkungen**
keine - nicht möglich	keine - festgelegt	auf : P 2.1

2.1.3 Technik

Technische Faktoren		
T 1.0	**Betriebssystem**	
T 1.01	**Windows / Linux**	
Flexibilität	**Änderbarkeit**	**Auswirkungen**
hoch - durch Java VM	hoch - variabel	auf : T 1.1, T 1.2, T 1.4

Technische Faktoren		
T 1.1	**Prozessor**	
T 1.11	**min. 1,6 Ghz**	
Flexibilität	**Änderbarkeit**	**Auswirkungen**
mittel - CPU´s entwickeln sich	mittel - solange vom Betriebssystem unterstützt	auf : T 1.0

Technische Faktoren		
T 1.2	**Arbeitsspeicher**	
T 1.21	**min. 1 GB**	
Flexibilität	**Änderbarkeit**	**Auswirkungen**
mittel - RAM werden größer	hoch - Änderung ist möglich	auf : keine

Technische Faktoren		
T 1.3	**Festplatte**	
T 1.31	**min. 10 GB**	
Flexibilität	**Änderbarkeit**	**Auswirkungen**
mittel - Mindestanforderung s. Anforderungsspezifikation	hoch - Änderung ist möglich	auf : T 1.0

Technische Faktoren		
T 1.4	**Umgebung**	
T 1.41	**JAVA VM**	
Flexibilität	**Änderbarkeit**	**Auswirkungen**
gering - festgelegt	gering - keine Kompatibilität garantiert	auf : T 1.0

2.2 Probleme und Strategien

(bearbeitet von: Yasin Ünsal)

Im folgenden werden mögliche Probleme beschrieben die bei der Entwicklung auftreten können und welche Faktoren damit verbunden sind. Ausserdem werden Lösungsmöglicheiten vorgestellt.

Unterschätzter Aufwand einer Aufgabe
Der Aufwand einer Aufgabe wurde von einem Mitarbeiter unterschätzt. Die Aufgabe ist nicht wie vorgesehen zu erledigen. **Einflussfaktoren** *OR 1.1 : Organisation Projektgruppe.* *OR 2.0 : Anforderungen an das Personal.* *OR 4.1 : Zeitplan.*
Lösung **Strategie S1: *Einteilen weiterer Mitarbeiter*** Die Aufgabe wird von mehreren Mitarbeitern bearbeitet und gelöst. **Strategie S2: *Aufsplitten der Aufgaben*** Die Aufgabe wird in kleineren Aufgaben geteilt und Mitarbeitern zugeordnet.

Abgabetermin
Da der Abgabetermin festgelegt und nicht veränderbar ist, ist es möglich dass nicht alle Aufgaben erfüllt werden können. **Einflussfaktoren** *OR 4.0 : Time to Market.* *OR 5.0 : Anzahl der Mitarbeiter.*
Lösung **Strategie S1: *Einsetzen zusätzlicher Mitarbeiter*** Um den Abgabetermin einhalten zu können, werden externe Mitarbeiter hinzugezogen.

Team interne Probleme
Dauerhaftes Fehlen von Mitarbeitern. **Einflussfaktoren** *OR 1.1 : Organisation Projektgruppe.* *OR 4.0 : Time to Market.* *OR 4.1 : Zeitplan.* *OR 5.0 : Anzahl der Mitarbeiter.*
Lösung **Strategie S1: *Reduzieren der Mitgliederanzahl*** Das Team arbeitet in kleinerer Gruppe weiter. **Strategie S2: *Ersetzen der Mitarbeiter*** Es wird mit einem Kompetenteren Mitarbeiter weiter gearbeitet.

Mangelnde Implementierungskenntnisse
Mitarbeiter die nicht qualifiziert genug sind, die Implementierungen durchzuführen. **Einflussfaktoren** *OR 1.1 : Organisation Projektgruppe.* *OR 4.1 : Zeitplan.* *OR 5.0 : Anzahl der Mitarbeiter.*
Lösung **Strategie S1: *Weiterbildung*** Die betroffenen Mitarbeiter werden durch Kompetentere Mitarbeiter geschult. **Strategie S2: *Ergänzung mit Fachpersonal*** Zur Hilfe werden zusätzliche Fachkräfte hinzugezogen.

Software Fehlfunktionen
Das Produkt funktioniert nicht Korrekt. **Einflussfaktoren** *P 2.0 : Benutzerschnittstelle.* *P 2.1 : Fehlererkennung.* *P 2.2 : Sicherheit.*
Lösung **Strategie S1: *Updates*** Die Fehler werden gefixt und per Updates dem Kunden zur verfügung gestellt.

3 Konzeptionelle Sicht

(bearbeitet von: Sascha Schmidt und Artur Malek; korrigiert von: Artur Malek)

Im Folgenden geben wir einen groben Überblick über die Komponenten und deren Zusammenarbeit. Anschließend gehen wir in der Modulsicht detailliert auf die einzelnen Komponenten und die beinhaltenden Module ein.

Das Gesamtsystem unterteilt sich in folgende Komponenten:

- Client
- Server
- Database
-XML Interface

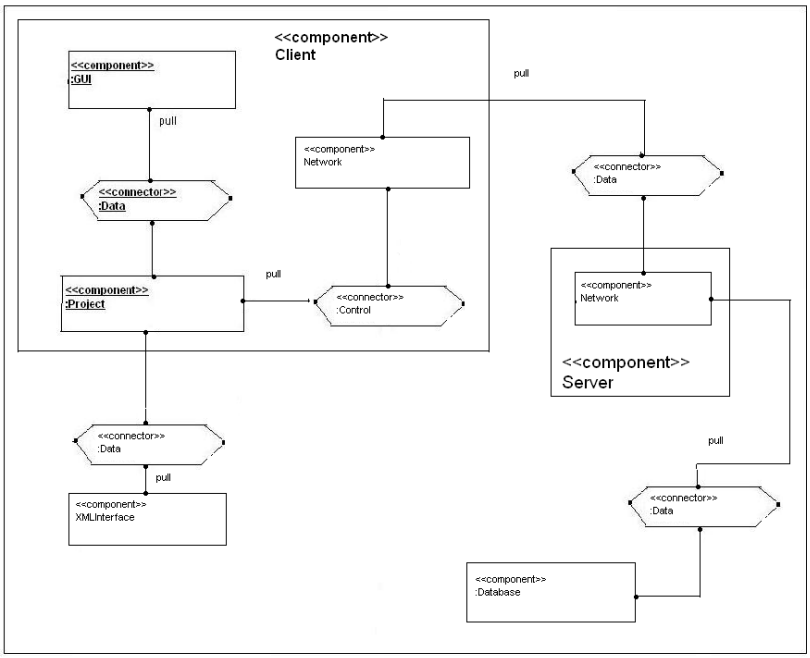


Abbildung 1: Konzeptionelle Sicht des Gesamtsystems

1. Client:
Der Client ist der Rechner, auf dem das System ausgeführt wird. Die Komponente Client besteht aus den Unterkomponenten GUI, Project und Network. Die GUI ist die grafische Benutzeroberfläche, die es dem Benutzer ermöglicht mit dem System

zu interagieren. Über sie sind alle Funktionen unseres Systems nutzbar. Die Komponente Project enthält die ganze Logik des Systems. Hier stehen alle Funktionen und Methoden unseres Systems. Ohne diese Komponente würde unser System nicht funktionieren. Die Komponente Network wird gebraucht, um die Kommunikation zwischen Client und Server zu gewährleisten. Die Kommunikation findet mit dem TCP/IP Protokoll statt.

2. Server:
Die Komponente Server wird gebraucht, um die Daten die vom Client angefordert werden, an die Datenbank zu übertragen. Daher besitzt auch der Server die Komponente Network für eine flüssige Kommunikation zwischen Server und Client. Um Daten von der Datenbank abzufragen, muss der Client sich mit dem Server verbinden. Dies geschieht durch Eingabe von Benutzername und Kennwort. Erst dann ist es dem Benutzer möglich, über den Server Daten von der Datenbank anzufordern oder Datan auf der Datenbank zu erstellen oder zu manipulieren.

3. Database:
Die Komponente Database wird für die Datenhaltung benötigt. Hier werden alle projektspezifischen Dateien abgespeichert. Das Speichern oder Bearbeiten von Daten ist, wie oben erwähnt, nur möglich, wenn man eine Verbindung zum Server aufgebaut hat.

4. XML Interface:
Die Komponente XML Interface ist dazu da, um ein Projekt lokal speichern zu können(also ohne Verbindung mit dem Server). Dabei werden alle projektspezifischen Daten in einer XML-Datei gespeichert und liegen dann auf dem Client-Rechner. Es ist auch möglich, ein lokal gespeichertes Projekt zu laden. Will man die lokal gespeicherten Daten auf die Datenbank speichern, so wird die XML-Datei ausgelesen und dementsprechend transformiert, sodass die Daten (nach Herstellung einer Verbindung zum Server) auf der Datenbank gespeichert werden können.

Die Folgenden Konnektoren kommen dabei zum Einsatz:

1.Control
Der Control-Konnektor beschreibt den Kontrollfluss. Dabei übergibt der Sender die Kontrolle an den Empfänger.

2.Data
Der Data Konnektor regelt den Datenaustausch zwischen den Komponenten. Dabei übermittelt der Data Konnektor die Eingansparameter an die nachfolgende Komponente und übergibt auch die Kontrolle an diese. Anschließend werden die Ergebnisse empfangen und an die ursprüngliche Komponente zurückgeliefert.

4 Modulsicht

(bearbeitet von: Sascha Schmidt und Artur Malek; korrigiert von: Artur Malek)

Die Modulsicht beschreibt die statisch logische Struktur des Systems. Dies wird unter Verwendung von Modulen, Schichten, Subsystemen und Schnittstellen getan. Die Modulsicht ist hierarchisch aufgebaut. Dies bedeutet, das Module in Teilmodule zerlegt werden, bis sie ein Arbeitspaket darstellen, welches von einem Mitarbeiter bearbeitet werden kann. Die Beschreibung aller Module und Teilmodule muss präzise formuliert sein, damit es möglich ist, anhand dieser Beschreibung das Modul zu implementieren. Einige Module in unserem System greifen auf Schnittstellen zu. Die Implementierung dieser Schnittstellen bleibt dem Entwickler überlassen, und wird von uns nicht zu diesem Zeitpunkt beschrieben. Wichtig ist es jedoch, um eine fehlerfreie Implemntierung zu garantieren, dass der Entwickler die Schnittstellensignatur bei der genauen Implementierung der Schnittstelle einhält.

Die in der konzeptionellen Sicht erläuterten Komponenten bilden die Module unseres Systems, diese werden dann in Teilmodule zerlegt, so dann sie als Arbeitspkaete bearbeitet werden können. Die Module, die wir hier nennen, heissen GUI, Project, und XML Interface. Das Modul GUI wird im folgenden nicht genau erläutert, da diese per Drag und Drop mit der von uns benutzten IDE NetBeans erstellt wird. Der Code wird von NetBeans dazu automatisch generiert. Verwendet wird aber Swing. Die Module XML Interface und Project werden im folgenden genauer erläutet.

Alle Module des Systems wurden durch UML-Diagramme modelliert und beschrieben. Methoden, die Parameter übergeben bekommen, haben folgende Notation in unserem UML-Diagrammen: z.b. bla (aus bla1 : String). Dies würde heißen, das bla eine Variable bla1 vom Typen String übergeben bekommt. do() : String heisst, das diese Methode einen String zurückgibt.

4.1 ProjectManagement

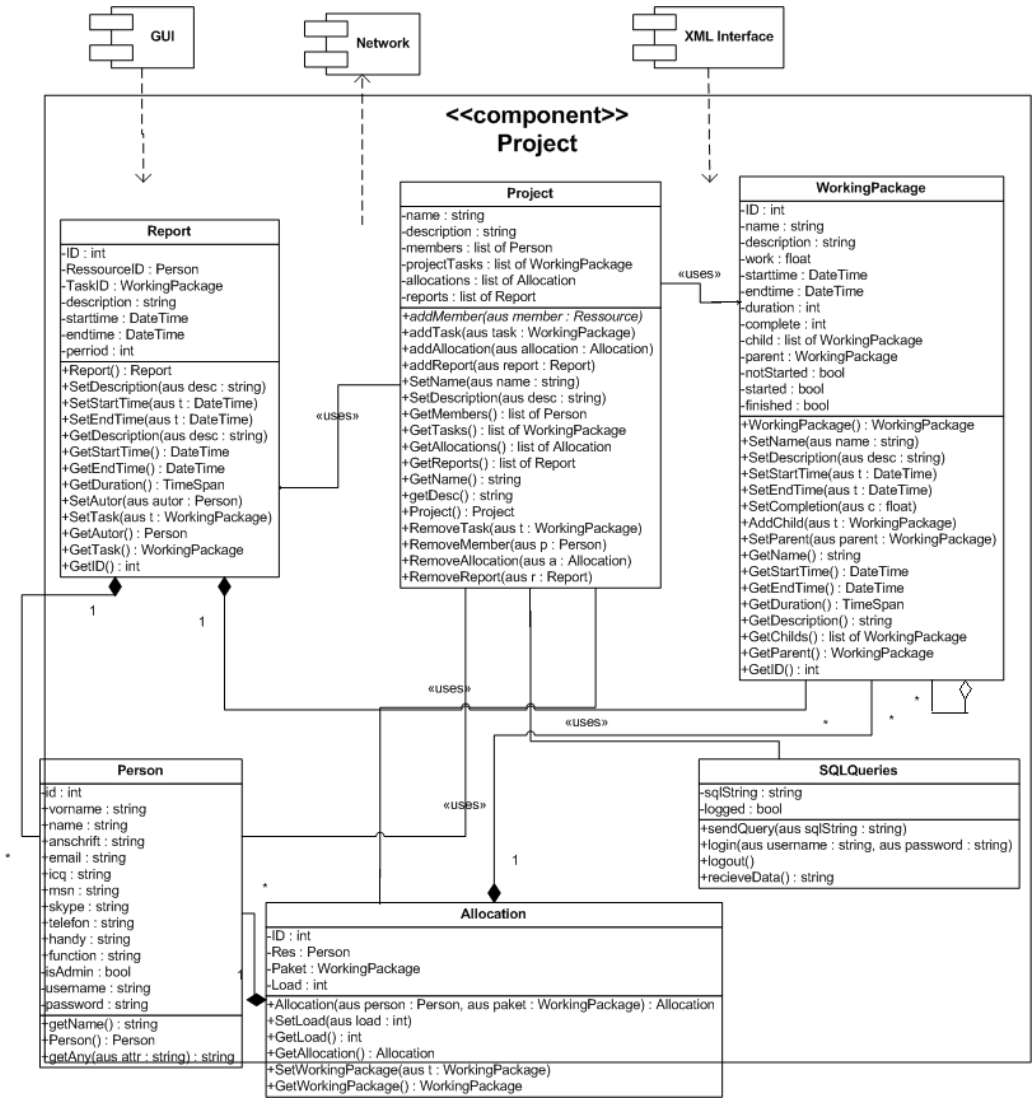


Abbildung 2: Das Modul ProjectManagement

Das Modul Project besteht aus den Teilmodulen WorkingPackage, Person, Allocation, Report,Project und SQLQueries. Diese Teilmodule besitzen Relationen untereinander. Zwischen den Modulen Person und Report herrscht eine Teil-von-Relation, eine Aggregation. Das heißt, dass ein Report nur dann erstellt werden kann, wenn eine Person vorhanden ist. Die gleiche Beziehung herrscht zwischen Person und Allocation. Zwischen dem Modul WorkingPackage und Allocation besteht eine Komposition, was auch eine Teil-von-Relation ist. Diese sagt aus, dass

nur dann eine Zuweisung existiert, wenn auch das Arbeitspaket dazu existiert. Die gleiche Beziehung herrscht zwischen den Modulen WorkingPackage und Report. Außerdem herrscht z.b. zwischen dem Modul Project und allen anderen Modulen eine «uses» Beziehung. Das Modul Project ist abhängig von dem Modul Connect-ToServer, da Daten die Project benötigt, erst angefordert werden können, wenn eine Verbindung zum Server hergestellt wurde. Vorher sind keine Abfragen der Datenbank möglich. Genauso ist das Modul GUI anhängig von Project, da eventuelle Veränderungen sich darauf auswirken, wie die GUI dies darzustellen hat. Im folgenden werden nun alle Teilmodule des Moduls Project beschrieben.

Project

Methode	Beschreibung
addMember(Person member)	Fügt dem Projekt eine neue Person hinzu
addWorkingPackage(WorkingPackage task)	Fügt dem Projekt ein Arbeitspaket hinzu
addAllocation(Allocation allocation)	Fügt dem Projekt eine Zuweisung einer Person zu einem Arbeitpaket in einem Projekt hinzu
addReport(Report report)	Fügt dem Projekt einen Bericht über ein Arbeitspaket hinzu
SetName(String name)	Bestimmt den Namen des Projekts
SetDesription(String desc)	Bestimmt die Beschreibung des Projekts
GetMembers()	Gibt eine Liste mit allen Personen zurück
GetWorkingPackages()	Gibt eine Liste mit allen Arbeitspaketen des Projekts zurück
GetAllocations()	Gibt eine Liste aller Zuweisungen der Arbeitspakete an Personen zurück
GetReports()	Gibt eine Liste aller Berichte zurück
GetName()	Gibt den Namen des Projekts zurück
getDesc()	Gibt die Beschreibung des Projekts zurück
Project()	Erzeugt ein neues Projekt
RemoveWorkingPackage(WorkingPackage wp)	Löscht ein Arbeitspaket aus dem Projekt
RemoveMember(Person p)	Löscht eine Person aus einem Projekt
RemoveAllocation(Allocation a)	Löscht eine Zuweisung aus einem Projekt
RemoveReport(Report r)	Löscht einen Bericht aus einem Projekt

Tabelle 1 : Modul Project

In dem Modul Project ist es möglich, ein neues Projekt anzulegen, was durch den Konstruktor Project() geschieht. In einem neu erzeugten Projekt kann man mittels addMember(Person member) dem Projekt eine neue Person hinzufügen.

Dazu muss als Parameter eine Person übergeben werden. Mittels addWorkingPackage(WorkingPackage wp) kann dem Projekt ein neues Arbeitspaket hinzugefügt werden. Als Parameter wird ein Arbeitspaket übergeben. Durch addAllocation(Allocation allocation) ist es möglich, dem Projekt Zuweisungen der Personen für Arbeitspakete hinzu zu fügen. Durch SetName(String name) und SetDescription(String desc) können der Name und die Beschreibung des Projekts bestimmt werden. Als Parameter werden die jeweiligen Strings verwendet, die dann dementsprechend den Namen oder die Beschreibung des Projekts beinhalten. Mit den Get-Methoden werden die jeweiligen Elemente , die angefordert wrrden, zurückgegeben. Mit den Remove-Methoden ist es möglich, die entsprechenden Elemente aus dem Projekt zu löschen.

Person

Methode	Beschreibung
getName()	Gibt Vor- und Nachnamen der Person zurück
Person()	Erzeugt eine neue Person
getAny(String attr)	Gibt ein bestimmtes Attribut einer Person zurück
getPerson(String attr)	Gibt bei Eingabe eines Attribut die dazugehörige Person zurück

Tabelle 2: Modul Person

Mit dem Modul Person können neue Personen, die an dem Projekt mitarbeiten erzeugt werden. Dies geschieht mit Hilfe des Konstruktors Person(). Alle zu einer Person dazugehörenden Informationen werden in den Feldern des Moduls gepeichert. Ausserdem verfügt das Modul Person noch über eine Methode getName(), die einen String bestehend aus dem Vor- und Nachmname der Person besteht. Die Methoden getPerson() und getAny() geben eine Person bzw. ein Attribut einer Person zurück. Mit dem Attribut isAdmin wird gesetzt, ob die jenige Person ein Administrator ist oder nicht. Ist isAdmin auf True gesetzt, so stehen dem Benutzer alle Funktionen des Systems zur Verfügung. Sollte sie auf False gesetzt sein, ist der Benutzer kein Administrator. In der GUI werden daraufhin einige Buttons und Menüpunkte auf Disable gesetzt, sprich der Benutzer kann diese nicht anklicken. Dadurch wird eine Unterscheidung zwischen Administrator und Normalbenutzter gewährleistet.

Methode	Beschreibung
WorkingPackage()	Erzeugt ein neues Arbeitspaket
SetName(String name)	Bestimmung des Namen des Arbeitspakets
SetDescription(String desc)	Bestimmung der Beschreibung des Arbeitspakets
SetStartTime(DateTime t)	Bestimmung der Startzeit der Bearbeitung des Arbeitpakets
SetEndTime(DateTime t)	Bestimmung der Endzeit der Bearbeitung des Arbeitspakets
SetCompletion(float c)	Bestimmung des Fortschritt des Arbeitspaketes
AddChild(WorkingPackage wp)	Hinzufügen eines Unterarbeitspakets zu einem Arbeitspaket
SetParent(WorkingPackage parent)	Bestimmung eines Eltern-Arbeitspakets
GetName()	Gibt den Namen eines Arbeitspakets zurück
GetStartTime()	Gibt die Startzeit eines Arbeitspakets zurück
GetEndTime()	Gibt die Endzeit eines Arbeitspakets zurück
GetDuration()	Gibt die Dauer der Bearbeitung eines Arbeitspakets zurück
GetDescription()	Gibt die Beschreibung eines Arbeitspakets zurück
GetChilds()	Gibt eine Liste der Unterarbeitspakete eines Arbeitspakets zurück
GetParent()	Gibt das Eltern-Arbeitspaket zurück
GetID()	Gibt die ID eines Arbeitspakets zurück

Tabelle 3: Modul WorkingPackage

Durch das Modul WorkingPackage wird ermöglicht, dass neue Arbeitspakete erstellt werden können. Hierfür wird der Konstruktor WorkingPackage() verwendet. Durch SetName(String name) und SetDescription(String desc) kann dem Arbeitspaket ein Name und eine Beschreibung hinzugefügt werden. Dafür werden der Name oder die Beschreibung als Parameter vom Typen String den jeweiligen Mehtoden übergeben. Druch die Methoden SetStartTime(DataTime t) und SetEndTime(DataTime t) können die Start- und die Endzeit der Bearbeitung des Arbeitspakets festgelegt werden. Dafür werden jeweils das Datum und die Uhrzeit als Parameter übergeben. Mit SetParent(WorkingPackage parent) ist es möglich, ein Eltern-Arbeitspaket zu definieren. Dieses kann z.b. die Anforderungsspezifikation sein. Mit AddChild(WorkingPackage wp) kann man dann Unterarbeitspakete von diesem Eltern-Arbeitspaket erstellen. Durch die Get-Methoden des Moduls WorkingPackage werden die Elemente zurückgeliefert, die gewünscht sind. Mit den Attributen notStarted, started und finished kann man festlegen, ob ein Arbeitspakete nocht nicht angefangen, angefangen oder beendet ist. Diese Attribute sind

vom Typen Boolean.

Report

Methode	Beschreibung
Report()	Erzeugt einen neuen Bericht
SetDescription(String desc)	Setzt den Inhalt des Berichts
SetStartTime(DataTime t)	Bestimmung der Startzeit des Arbeitspakets über das Bericht geschrieben wird
SetEndTime(Datatime t)	Bestimmung der Endzeit des Arbeitspakets über das Bericht geschrieben wird
GetDescription(String desc)	Gibt die Beschreibung eines Berichts zurück
GetStartTime()	Gibt die Startzeit eines Arbeitspakets zurück
GetEndTime()	Gibt die Endzeit eines Arbeitspakets zurück
GetDuration()	Gibt die Dauer der Bearbeitung eines Arbeitspakets zurück
SetAutor(Person autor)	Bestimmung des Autors des Berichts
SetWorkingPackage(WorkingPackage t)	Bestimmung des Arbeitspakets, über das Bericht geschrieben wird
GetAutor()	Gibt den Autor des Berichts zurück
GetWorkingPackage()	Gibt das Arbeitspaket, über das ein Bericht geschrieben wurde zurück
GetID()	Gibt die ID des Berichts zurück

Tabelle 4 : Modul Report

Das Modul Report erzeugt neue Berichte. Dies passiert durch den Konstruktor Report(). Mit SetDescription(String desc) wird der Inhalt des Berichts eingetragen. mit SetStartTime(DataTime t) und SetEndTime(DataTime t) werden die Start und Endzeit des Arbeitspakets, für das der Bericht geschrieben wird, festgelegt. Mit SetAutor(Person autor) wird bestimmt, welche Person diesen Bericht schreibt. SetWorkingPackage(WorkingPackage wp) legt fest, für welches Arbeitspaket dieser Bericht geschrieben wird. Die einzelnen Get-Methoden geben die dementsprechenden Elemente zurück, die durch den Aufruf gewüscht werden.

Allocation

Methode	Beschreibung
Allocation(Person person, WorkingPackage paket)	Erzeugt eine neue Zuweisung eines Arbeitspakets zu einer Person
SetLoad(int load)	Bestimmt die Auslastung der Person
GetLoad()	Gibt die Auslastung einer Person zurück
GetAllocation()	Gibt die Zuweisung eines Arbeitspakets zu einer Person zurück
SetWorkingPackage(WorkingPackage wp)	Bestimmung des Arbeitspakets, das einer Person zugeteilt werden soll
GetWorkingPackage()	Gibt ein Arbeitspaket zurück, das einer Person zugeteilt wurde

Tabelle 5 : Modul Allocation

Das Modul Allocation wird gebraucht, um Zuweisungen von Personen zu Arbeitspaketen zu realisieren. Eine neue Zuweisung wird mit Allocation(Person person, WorkingPackage paket), dem Konstruktor, erzeugt. Dafür muss dem Konstruktor die Betroffene Person und das Arbeitspaket übergeben werden. Mit SetLoad(int load) kann man die Auslastung einer Person festlegen. Dafür wird eine Zahl vom Typen Integer als Parameter übergeben. Mit den Get()-Methoden ist es möglich, Elemente, die zum Erstellen einer Zuweisung gebraucht werden, sich zurück zu geben zu lassen.

SQLQueries

Methode	Beschreibung
sendQuery(String sqlString)	Sendet eine Anfrage über den Server an die Datenbank und liefert die Daten an die Funktion, die sie benötigt
recieveData()	Gibt die Daten, die von der Datenbank geliefert wurden, an die jeweilige Funktion zurück
login(String username, String password)	Auf dem Server anmelden
logout() Abmelden auf dem Server	

Tabelle 6 : Modul SQLQueries

Das Modul SQLQueries dient dazu, die SQL-Anfragen über den Server an die Datenbank zu leiten. Mit der Funktion sendQuery() ist es möglich, Daten, die von der Datenbank gebraucht werden, anzufordern. Dazu muss man mit dem Server verbunden sein. Dies geschieht mit der Funktion login(). Ist man eingelogget, so wird

die Variable logged auf True gesetzt. Mit logout() kann man sich wieder vom Server abmelden. Ist dies geschehen, ist es nicht mehr mölgich, Daten von der Datenbank zu erhalten. mit recieveData() werden die Daten zu den jeweiligen Funktionen, die die Daten benötigen, zurückgegeben.

4.2 XML Interface

Mit XML Interfaceist es möglich, Projektdaten lokal als XML-Dateien zu speichern. Dies geschieht mit der Funktion saveProject(). Mit loadProject() kann man ein Projekt, das als XML-Datei gespeichert wurde, wieder öffnen und weiterbearbeiten. parseXMLFile() wird gebraucht, um die Daten aus den XML-Dateien auszulesen, damit sie korrekt im System dargestellt werden können.

Methode	Beschreibung
saveProject(Project p)	speichert ein Projekt als XML Datei
loadProject(String filename)	lädt ein Projekt welches als XML Datei gespeicher ist
parseXMLFile	liest eine XML-Datei ein um sie korrekt im System darstellen zu können

4.3 GUI

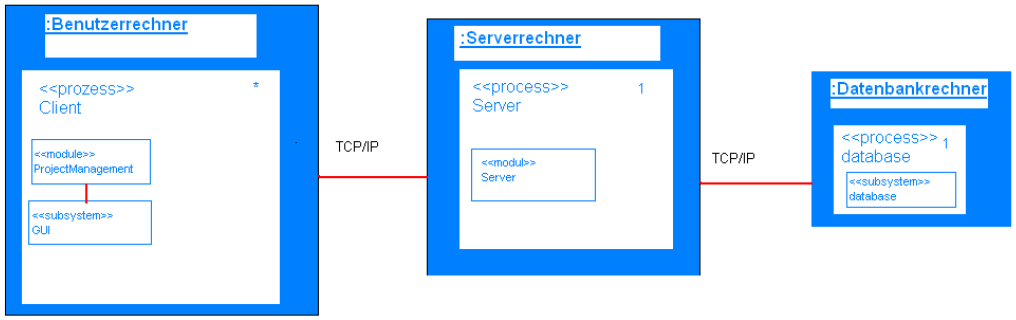
Das Modul GUI sorgt dafür, dass der Benutzer mit dem System interagieren kann. Es stellt alle Methoden zur Verfügung , die dazu gebraucht werden, um die vom Modul ProjectManagement vorbereiteten Informationen auf zu nehmen und diese so zu bearbeiten, das sie für den Benutzer grafisch auf dem Bildschirm dargestellt werden. Andererseits müssen Eingaben, die in der GUI getätigt werden, verarbeitet und an das Modul ProjectManagement zur Weiterverarbeitung übergeben werden. Damit dies funktioniert, müssen wir die Funktionen des Moduls ProjectManagement mit den Teilmodulen des Moduls GUI verbinden. Daher besteht eine Abhängigkeit zwischen den Modulen GUI und ProjectManagement. Das Modul GUI wird von uns nicht selbst implementiert, sondern wir benutzen Swing, was eine in Java intergrierte Schnittstelle zur Erstellung von grafischen Benutzeroberflächen ist. Daher wird hier nicht beschrieben, aus welchen Teilmodulen die GUI

besteht und welche Methoden sie zur Verfügung stellen, sondern nur, wie wird dieses Modul verwenden müssen.

5 Ausführungssicht

(bearbeitet von: Volkan Gizli; korrgiert von: Artur Malek)

In diesem Punkt wollen wir uns zur Ausführungssicht äussern. Dazu wird die Harmonie bzw. das Verhalten und die Kommunikation der einzelnen Komponenten zur Laufzeit beschrieben.
Ausserdem stellen wir dar, inwiefern die Laufzeitkomponenten an welcher Stelle im Gesamtsystem miteinander kommunizieren.
Dies geschieht folgendermassen:



In der Ausführungssicht wird das Laufzeitverhalten des Systems durch das Spezifizieren der Laufzeitelemente und die Kommunikation, die untereinander besteht, beschrieben.

Wie in der Modulsicht beschrieben, muss man sich erst auf dem Server anmelden, um Daten vom Datenbankrechner anzufordern. Ist man mit dem Server verbunden, dann wartet der Server auf eine Anfrage , die Daten vom Datenbankrechner abrufen will. Anfragen schickt der Client an den Server, und fordert vom Server Daten an oder schickt Daten an den Server. Der Server wertet die Anfragen aus, schaut in der Datenbank und gibt die Daten zurück, die gefordert wurden, oder fügt neue Daten hinzu oder überschreibt schon vorhandene Daten. Client und Server und Server und die Datenbank kommunizieren jeweils per TCP/IP.

Damit das System betriebsfähig ist, sind 3 Prozesse nötig:
Der Client, der Server und der Datenbankserver. Es können beliebig viele Clients mit dem Server kommunizieren bzw. das System nutzen. Es wird aber nur ein Server und ein Datenbankserver benötigt. Alle 3 Elemente können auf verschiedenen Rechner installiert worden sein. Hauptelement des Clients ist die GUI, da über sie das ganze System gesteuert wird. Der Server und die Datenbank benötigen keine GUI.

6 Evolution

(bearbeitet von: Volkan Gizli)

In diesem Punkt wollen wir auf die Vorgehensweisen eingehen, die bei möglichen Expandierungen in der Software notwendig sind.

6.1 Erweiterungen der GUI's

Damit man in der Zukunft nach Belieben des individuellen Beutzers unterschiedliche GUI's verwenden kann, ist es nötig, die Software mit einer Funktionalität zu implementieren, die die Auswahl weiterer Oberflächen realisiert.
Diese Funktionalität muss so aufgebaut sein, so dass sie die entsprechenden Elemente, die in der GUI vom Style geändert werden sollen, mit entpsrechenden Bilddateien referenziert. Die textuellen Elemente könnte man mit einer XML inkludieren. Dafür könnte man zum Beispiel direkt einen Ordner beziehen, der alle Elemente für einen beliebigen Style beinhaltet oder aber bei Linux zum Beispiel mit dem Importieren einer tar-Datei, die alle Ressourcen beinhaltet.

6.2 Synchronisation mit einem PDA

Die Möglichkeit der Synchronisation mit einem PDA wäre in der Zukunft auch realisierbar. Dafür bräuchte man an sich nur eine adequate GUI, die für ein PDA geeignet ist. Die Funktionen bzw. Klassen könnte man dann entsprechend mit implementieren.
Ausserdem ist noch eine Änderung bzw. zusätzliche Funktion für die Schnittstelle in der Software notwendig.

6.3 Erweiterungen der Sprachfunktionen

Um Benutzern die Benutzung der Software zu vereinfachen, kann man in der Zukunft eine Erweiterung bzw. Funktion für die Umstellung von weiteren Sprachen zur Verfügung stellen. Die Realisierung dieser Funktion könnte genau so ähnlich erfolgen, wie die der Inkludierung der textuellen Elemente in der GUI-Umstellung. Die Referenzierung mit einer entsprechenden XML-Datei, die die zu inkludierenden Begriffe beinhaltet, wäre somit eine gute geeignete Möglichkeit.

6.4 Rechtschreibüberprüfungsfunktion

Damit man keine Fehler in der Rechtschreibung begeht, kann man in der Zukunft die Funktion zur Rechtschreibüberprüfung einbauen.
Diese Funktion muss so realisiert werden, so dass sie die Strings mit einem vorgegebenem Container voller Stringmuster vergleicht.
Stimmt dieser String mit einem Muster überein, so ist die Rechtschreibung richtig.
Stimmt sie jedoch nicht überein, so kann man den String als fehlerhaft zum Beispiel markierend hervorhebend darstellen.
Bei möglichen Rechtschreibänderungen bzw. neuen Wortschöpfungen in der Sprache sollte der entsprechende Container aktualisierbar sein.

6.5 Erstellung von Backups

Damit die Daten bei Konflikten durch Backups sichergestellt sind, wäre in der Zukunft eine Backup-Funktion für die Erstellung von Sicherheitskopien notwendig zu implementieren.
Diese Backupfunktion wäre durch die SQL-Klausel **SELECT INTO** relativ gut zu realisieren, in dem sie die entsprechenden Tabellen inkrementell sichert.